Using the Intel Distribution for Python

Intel Distribution for Python provides accelerated performance for numerical computing and data science on Intel architectures. With the distribution, math and statistical packages such as NumPy, SciPy, and scikit-learn are linked with Intel's performance libraries for near-native code speeds. Libraries include:

- Math Kernel Library (Intel MKL) with optimized BLAS, LAPACK, FFT, and random number generators
- Message Passing Interface (Intel MPI)
- Thread Building Blocks (Intel TBB)
- Data Analytics Acceleration Library (Intel DAAL)

In general, you do not need to change your Python code to take advantage of the improved performance Intel's Python Distribution provides. However, for random number generators, we recommend using the MKL-based random number generator **`numpy.random_intel`** as a drop-in replacement for **`numpy.random`**.

For machine learning, Intel Distribution for Python provides deep learning software such as Caffe and Theano, as well as classic machine learning libraries, such as scikit-learn and pyDAAL (which implements Python bindings to Intel DAAL).

Accessing Intel Python on Pleiades

On Pleiades, you can access a full standalone version of Intel Distribution for Python via a module in the **`/nasa`** directory.

```
pfe% module load comp-intel/2018.3.222
pfe% module load python3/Intel_Python_3.6_2018.3.222
pfe% which python
/u/scicon/tools/opt/sles12/python/2018.3.222/intelpython3/bin/python
pfe% python --version
Python 3.6.3 :: Intel Corporation
```

Note: Because the Python 2 end-of-life date is 2020, only the Intel Python 3 distribution is made available on Pleiades to encourage Python 2 users to migrate to Python 3.

To see what packages are included in this distribution, run:

```
pfe% conda list
```

WARNING: Do not load any other modules if you don't need them in your working session, as there are potential software conflicts between the Intel Python and other modules such as Tecplot, Python from other distributions, and various MPI libraries.

If You Need Additional Packages

Only NAS system administrators can add packages to the **`/nasa`** or **`/u/scicon`** directory, where this Intel Python module is installed. If you need additional packages that are not currently included, and you think the packages might be widely used by other Pleiades users, send a request to support@nas.nasa.gov. Otherwise, follow the instructions below to add packages to your own directories.

Installing Packages from the Python Package Index

If the package is available at The Python Package Index (PyPI), use **pip** to install it in your **$HOME/.local/lib/python3.6/site-packages** directory, where Intel Python will be able to find the package.

To install a package, run:

```
pfe% which pip
/u/scicon/tools/opt/sles12/python/2018.3.222/intelpython3/bin/pip
pfe% pip install --user package_name
```

Note: The virtualenv tool for managing Python environments is not available in the Intel Distribution for Python.

Installing Packages from the Anaconda Repositories

If the package is available at Anaconda Repositories, use the **conda** tool to install it.

Note: Unlike **pip install**, there is no **--user** option for **conda install**. Instead, you must create a new Conda enviroment where all the packages you need are available, including the Intel Distribution for Python.

Use one of the following two methods to create a new Conda environment and install packages.

Method 1: Clone the Intel Python from the /u/scicon Directory

Clone the entire Intel Python Distribution from **/u/scicon** to your **$HOME/.conda/envs** directory and add the packages to the new environment:

```
pfe% conda create -n my_clone_env --clone="/u/scicon/tools/opt/sles12/python/2018.3.222/intelpython3"
pfe% conda install -n my_clone_env package_name
```

Note: Cloning the entire distribution requires ~3.6 GB of space in your $HOME directory.

When you want to use the **my_clone_env** environment, you will need to activate that environment under **bash**, because **csh** is not supported by **conda**. Remember to deactivate when you are done using that environment.

```
pfe% bash
pfe% source activate my_clone_env
(my_clone_env) bash$ which python
/homeX/username/.conda/envs/my_clone_env/bin/python
(my_clone_env) bash$ #do your work
(my_clone_env) bash$ source deactivate
```

Method 2: Install Your Own Anaconda or Miniconda Distribution

Instead of using the Intel Python provided in the **/u/scicon** directory, you can install your own Anaconda or Miniconda distribution with the packages you want, and follow the steps in the document Installing IntelÂ® Distribution for Python and Intel Performance Libraries with Anaconda to add some of the packages from the distribution.

To learn more about using **conda**, see **conda -h** or download the Conda Cheat Sheet.

Additional Resources

- Intel Distribution for Python Documentation
- Intel Distribution fo Python - Highlights & Overview
- Intel Distribution for Python - Build for Speed
- Techniques Used to Accelerate Performance of numpy and scipy in Intel Distribution for Python
- Achieving High-Performance Computing with the Intel Distribution for Python

---

Article ID: 562
Last updated: 08 Dec, 2020
Revision: 63
Filesystems & Software -> Software -> Using the Intel Distribution for Python
https://www.nas.nasa.gov/hecc/support/kb/entry/562/